

**ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 6**  
**«ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ И ИХ ОПИСАНИЕ СРЕДСТВАМИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ»**

**Цель работы:** изучить алгоритмические конструкции и их описание, научиться составлять алгоритмы в виде блок-схем и на псевдокоде.

**Теоретические сведения к практической работе**

Алгоритм является фундаментальным понятием информатики. Представление о нем необходимо для эффективного применения вычислительной техники к решению практических задач.

**Алгоритм** - это последовательность действий, которая должна быть выполнена для достижения желаемого результата.

**Алгоритм решения некоторой задачи** - это алгоритм, приводящий к решению этой задачи за конечное число действий

**Свойства алгоритма и его исполнители**

**1. Дискретность.**

Разделение алгоритма на последовательность законченных действий – шагов. Каждое действие должно быть закончено прежде, чем исполнитель приступит к выполнению следующего шага.

**2. Результативность.**

Получение из исходных данных результата за конечное число шагов.

**3. Массовость.**

Возможность применения алгоритма к большому количеству различных исходных данных.

**4. Детерминированность.**

Выполнение команд алгоритма в строго определенной последовательности.

**5. Выполнимость и понятность.**

Алгоритм не должен содержать предписаний, смысл которых может восприниматься неоднозначно.

**6. Точность.**

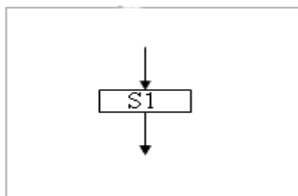
Запись алгоритма должна быть такой, чтобы на каждом шаге его выполнения было известно, какую команду нужно выполнять следующей.

**7. Конечность.**

Завершение работы алгоритма за конечное число шагов.

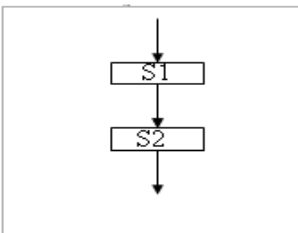
Наиболее понятно структуру алгоритма можно представить с помощью блок-схемы, в которой используются геометрические фигуры (блоки), соединенные между собой стрелками, указывающими направление потоков информации (последовательность выполнения действий). Приняты определенные стандарты графических изображений блоков. Например, команду обработки информации помещают в блок, имеющий вид прямоугольника, проверку условий - в ромб, команды ввода или вывода - в параллелограмм, а овалом обозначают начало и конец алгоритма.

Структурной элементарной единицей алгоритма является простая команда, обозначающая один элементарный шаг переработки или отображения информации. Простая команда на языке схем изображается в виде функционального блока.



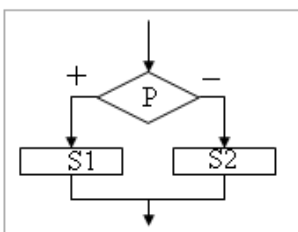
Данный блок имеет *один вход* и *один выход*. Из простых команд и проверки условий образуются составные команды, имеющие более сложную структуру и тоже *один вход* и *один выход*.

Структурный подход к разработке алгоритмов определяет использование только базовых алгоритмических структур (конструкций): следование, ветвление, повторение, которые должны быть оформлены стандартным образом.

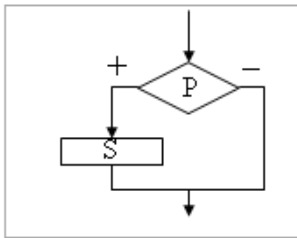


Рассмотрим основные структуры алгоритма.

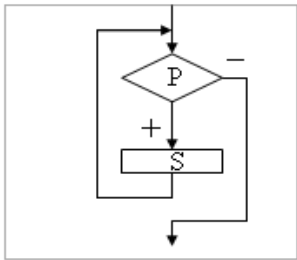
Алгоритм *следования* состоит только из простых команд. На рисунке простые команды имеют условное обозначение *S1* и *S2*. Из команд следования образуются линейные алгоритмы. Примером линейного алгоритма будет нахождение суммы двух чисел, введенных с клавиатуры.



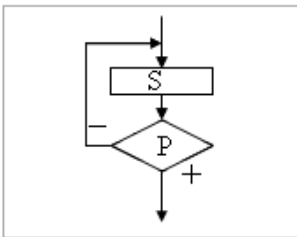
Алгоритм *ветвления* - это составная команда алгоритма, в которой в зависимости от условия *P* выполняется или одно *S1*, или другое *S2* действие. Из команд следования и команд ветвления составляются разветвляющиеся алгоритмы (алгоритмы ветвления). Примером разветвляющегося алгоритма будет нахождение большего из двух чисел, введенных с клавиатуры.



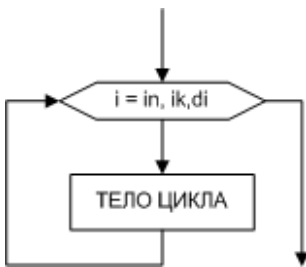
Алгоритм ветвления может быть полной и неполной формы. Неполная форма используется тогда, когда необходимо выполнять действие  $S$  только в случае соблюдения условия  $P$ . Если условие  $P$  не соблюдается, то команда ветвления завершает свою работу без выполнения действия. Примером неполной формы ветвления будет уменьшение в два раза только четного числа.



Алгоритм *повторения* - это составная команда алгоритма, в которой в зависимости от условия  $P$  возможно многократное выполнение действия  $S$  (тело цикла). Из команд следования и команд повторения составляются циклические алгоритмы (алгоритмы повторения). На рисунке представлен циклический алгоритм с предусловием. Называется он так потому, что вначале проверяется условие, а уже затем выполняется действие (тело цикла). Причем действие выполняется, пока условие верно. Пример циклического алгоритма может быть следующий. Пока с клавиатуры вводятся положительные числа, алгоритм выполняет нахождение их суммы.



В алгоритме повторения с постусловием сначала выполняется действие  $S$  (тело цикла) и лишь затем, проверяется условие  $P$ . Причем действие повторяется до тех пор, пока условие ложно. Примером будет уменьшение положительного числа до тех пор, пока оно неотрицательное. Как только число становится отрицательным, алгоритм повторения заканчивает свою работу.



Безусловный циклический алгоритм, его удобно использовать, если известно, сколько раз необходимо выполнить тело цикла. Выполнение безусловного циклического алгоритма начинается с присвоения переменной  $i$  стартового значения  $in$ . Затем следует проверка, не превосходит ли переменная  $i$  конечное значение  $ik$ . Если превосходит, то цикл считается завершенным, и управление передается следующему за телом цикла оператору. В противном случае выполняется тело цикла, и переменная  $i$  меняет свое значение в соответствии с указанным шагом  $di$ . Далее снова производится проверка значения переменной  $i$  и алгоритм повторяется. Переменную  $i$  называют параметром цикла, так как это переменная, которая изменяется внутри цикла по определенному закону и влияет на его окончание.

С помощью соединения только этих элементарных конструкций (последовательно или вложением) можно "собрать" алгоритм любой степени сложности.

## Способы описания алгоритма

**Задача: Найти сумму двух чисел 2 и 3**

### 1. Словесный способ

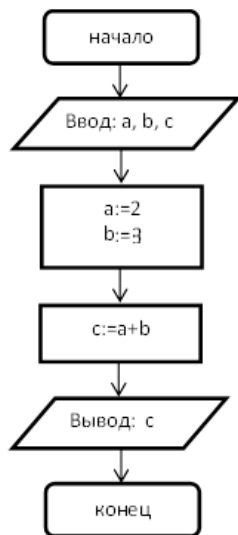
Алгоритм представляет собой описание на естественном языке последовательных этапов обработки данных.

К двум прибавляем три получаем пять.

### 2. Графический способ

Изображение алгоритма в виде последовательности связанных между собой функциональных блоков.

Блок-схема позволяет сделать алгоритм более наглядным и выделяет в алгоритме основные алгоритмические структуры (линейная, ветвление, выбор и цикл).



Элемент блок-схемы	Назначение элемента блок-схемы
	Прямоугольник с закругленными углами, применяется для обозначения начала или конца алгоритма
	Параллелограмм, предназначен для описания ввода или вывода данных, имеет один вход сверху и один выход внизу
	Прямоугольник, применяется для описания линейной последовательности команд, имеет один вход сверху и один выход внизу
	Ромб, служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход сверху и два выхода (налево, если условие выполняется, и направо, если условие не выполняется)

### 3. Псевдокод

Система обозначений и правил, предназначенная для единообразной записи алгоритмов.

Алг Сумма

дано a, b, c;

надо c=a+b;

нач a:=2, b:=3;

c:= a+b;

кон.

### 4. Программный способ (алгоритмический)

Алгоритм, предназначенный для записи на компьютере, должен быть записан на понятном ему языке. Такой язык называется **языком программирования**, а запись алгоритма на этом языке – **программа**.

Языки программирования предназначены для создания программ, которые могут быть исполнены ЭВМ или другими автоматическими устройствами, например, станками с числовым программным управлением. Система *Pascal ABC* предназначена для обучения программированию на языке Паскаль. Как и любой алгоритм, являющийся последовательностью инструкций, программа на языке Паскаль состоит из команд (операторов), записанных в определенном порядке и формате. Команды позволяют получать, сохранять и обрабатывать данные различных типов (например, целые числа, символы, строки символов, т.д.).

Кроме команд в записи программы участвуют еще так называемые "служебные слова", организующие структуру программы. Правила языка Паскаль предусматривают единую для всех программ форму основной структуры

```

Program <Имя программы>;
<Раздел описаний>;
Begin
<Тело программы>;
End.
  
```

Здесь слова **Program**, **Begin** и **End** являются служебными. Правильное и уместное употребление этих слов является обязательным.

**Пример.** Ввести в компьютер два целых числа, найти их сумму, результат вывести на экран с поясняющим текстом.

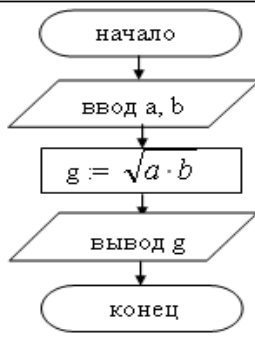
**Внимание!** Две косые черты (//) отделяют комментарии, их набирать не нужно.

```

program raschet;// название программы
uses crt;// подключаемые модули
var x, y, s:integer;// объявление имен переменных и их типа
begin// начало исполнительной части
writeln('Введите два целых числа');//написать на экране текст
readln(x,y);//прочитать данные с клавиатуры и запомнить их в переменных
s:=x+y; // выполнить расчет и запомнить его в переменной
writeln('Сумма чисел =',s); //написать на экране текст и значение переменной
end. //конец программы
  
```

### Линейный алгоритм

Пример записи алгоритма в виде блок-схемы, псевдокоде и на языке Паскаль.

Блок-схема	Псевдокоды	Паскаль
	<p><b>алг</b> среднее геометрическое</p> <p><b>вещ</b> a, b, g</p> <p><b>нач</b></p> <p><b>ввод</b> a, b</p> <p><math>g := (a * b) ^ (1/2)</math></p> <p><b>вывод</b> g</p> <p><b>кон</b></p>	<pre> <b>program</b> Srednee_geometr;   <b>var</b> a, b, g: real; <b>begin</b>   <b>readln</b> (a, b);   s := sqrt(a * b);   <b>writeln</b> (g) <b>end.</b>                     </pre>

### ХОД РАБОТЫ:

#### Задание № 1

Некий злоумышленник выдал следующий алгоритм за алгоритм получения кипятка:

- 1 Налить в чайник воду.
- 2 Открыть кран газовой горелки.
- 3 Поставить чайник на плиту.
- 4 Ждать, пока не закипит вода.
- 5 Поднести спичку к горелке.
- 6 Зажечь спичку.
- 7 Выключить газ.

Исправьте алгоритм, чтобы предотвратить несчастный случай.

#### Задание № 2

Имеются два кувшина емкостью 3 л и 8 л. Напишите алгоритм на естественном языке, выполняя который можно набрать из реки 7 л воды. (Разрешается пользоваться только этими кувшинами.)

#### Задание № 3

Построить блок-схему линейного алгоритма вычисления значения выражения по вариантам и написать программу на псевдокоде.

<b>Вариант 1</b> $z = ctg \left( \frac{5}{4} \pi + \frac{3}{2} \alpha \right)$	<b>Вариант 5</b> $z = tg 3\alpha$	<b>Вариант 9</b> $z = 2 \sin \alpha$
<b>Вариант 2</b> $z = \frac{\cos \alpha + \sin \alpha}{\cos \alpha - \sin \alpha}$	<b>Вариант 6</b> $z = \frac{1}{\sqrt{b+2}}$	<b>Вариант 10</b> $z = \frac{\sqrt{m} - \sqrt{n}}{m}$
<b>Вариант 3</b> $z = \cos^2 \alpha + \cos^4 \alpha$	<b>Вариант 7</b> $z = \frac{4 - a^2}{2}$	<b>Вариант 11</b> $z = -\sqrt{m}$
<b>Вариант 4</b> $z = \frac{1 - tg \alpha}{1 + tg \alpha}$	<b>Вариант 8</b> $z = \frac{1}{\sqrt{a} + \sqrt{2}}$	<b>Вариант 12</b> $z = \sqrt{\frac{x+3}{x-3}}$

#### Задание № 4

Перед выходным днем папа сказал своему сыну: «Давай спланируем свой завтрашний день. Если будет хорошая погода, то проведем день в лесу. Если же погода будет плохая, то сначала займемся уборкой квартиры, а во второй половине дня сходим в зоопарк». Что получится на выходе блок-схемы, если:

- а) погода хорошая;
- б) погода плохая?

#### Задание № 5

Сделать вывод о проделанной практической работе

---



---



---